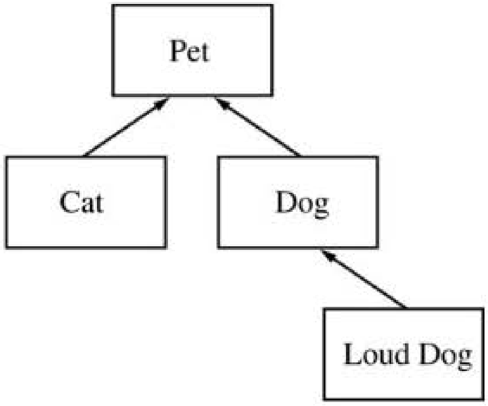
Inheritance & Arrays: Pet FRQ

Consider the hierarchy of classes shown in the following diagram.



Note that a *Cat* "is-a" *Pet*, a *Dog* "is-a" *Pet*, and a *LoudDog* "is-a" *Dog*.
Each *Pet* has a name that is specified when it is constructed.

```
public class Pet
{
     private String myName;

     public Pet(String name)
     {  myName = name;  }

     public String getName()
     {  return myName;  }

     public String speak()
     { /* implementation not shown */ }

}
```

(a) Given the class hierarchy shown above, write complete class declarations for the classes *Dog* and *Cat*, including implementations of their constructors and methods. The *Dog* method *speak()* returns "*woof*" when it is invoked. The *Cat* method *speak()* returns "*meow*" when it is invoked.

(b) Assume that class `Dog` has been declared as shown at the beginning of the question. If the `String` `dogSound` is returned by the `Dog` method `speak()`,then the `LoudDog` method `speak()` returns a `String` containing `dogSound` repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class `LoudDog`, including implementations of its constructor and method(s).

(c) Consider the following partial declaration of class `Kennel`.

```
public class Kennel
{
     // all elements are references to Pet objects
     private Pet[] petList;

     // postcondition: for each Pet in the kennel, its name
     //                followed by the result of a call to
     //                its speak method has been printed,
     //                one line per Pet
     public String allSpeak()
     {  /* to be implemented in this part */  }

     // ... constructor and other methods not shown
}
```

Write the `Kennel` method `allSpeak()`. For each `Pet` in the kennel, `allSpeak()` returns a `String` with the name of each `Pet` followed by the result of a call to its `speak()` method, one line per `Pet`.

In writing `allSpeak()`, you may use any of the methods defined for any of the classes specified for this problem. Assume that these methods work as specified, regardless of what you wrote in parts *(a)* and *(b)*. Solutions that reimplement functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method allSpeak below.

```
// postcondition: for each Pet in the kennel, its name
//                followed by the result of a call to its
//                has been returned, one line per Pet
public String allSpeak()
```